

What is claimed is:

Sub All

1. A method for querying stored multimedia data in a computer system,
comprising:

5 receiving into an intermediate level a high-level concept from a user
describing data to be retrieved;

translating, in said intermediate level, said high-level concept into
low-level queries by using system pre-defined high-level concepts; and

transferring said low-level queries to a low level comprising one or more
10 search engines; said one or more search engines performing a query of the stored
multimedia information using said low-level queries.

2. The method of claim 1 wherein said intermediate level comprises:

a set of library modules, said set of library modules comprising:

15 a concept library module for storing concepts;

one or more library modules adapted to store said data from said one or
more data sources;

a cataloger module adapted to construct a new concept from said
high-level concept using data from said concept library and library modules, thereby
20 creating a concept construct, and to pass said concept construct to said concept library
module for storage as a concept; and

an interpreter module adapted to translate said high-level concept into low-level queries using said concepts stored in said construct library and to pass said low-level queries to said one or more search engines.

5 3. The method of claim 2 wherein said set of library modules further comprises at least one library module selected from the group comprising:

 a feature library module adapted to store multimedia features;

 a matching algorithm library module adapted to store matching algorithms; and

10 a constraint library module adapted to store feature constraints.

 4. The method of claim 3 wherein each said library module further comprises an application program interface to receive said data from a said data source.

15 5. The method of claim 3 wherein said cataloger module further performs the steps of:

 selecting a set of concept features from said feature library module;

 selecting a set of concepts from said concept library module for use as child-concepts; and

20 selecting a set of constraints on said child concepts from said constraint library module.

6. The method of claim 1 wherein said each said concept comprises a triplet of a set of child-concepts, a set of features, and a set of relationships.

7. The method of claim 6 wherein said concepts comprise a hierarchical fuzzy graph data tree-structure comprising nodes, aggregation edges, and association edges and wherein:

said nodes correspond to said concepts and said features;

said aggregation edges correspond to parent-child relationships; and

said association edges correspond to said constraints.

8. The method of claim 7 wherein said edges are weighted.

9. The method of claim 3 further comprising a matching algorithm comprising GetNextMatch(), AssignNextMatch(), and ShiftNextMatch() procedures, wherein:

said GetNextMatch() procedure comprises the steps:

testqueue: if *queue*.Empty();

return *NULL*;

head --> *queue*.Pop();

if *head*.Complete();

return *head*;

head2 --> *head*.Copy();

head2.AssignNextMatch();

if *head2*.Valid();

queue.Push(*head2*);

head.ShiftNextMatch();

queue.Push(*head*);

5 Goto testqueue;

said AssignNextMatch() procedure comprises the steps:

child --> GetNextUnassigned();

child.match_ptr++;

if (*child.match_ptr* == NULL), then;

10 *child.match_ptr* --> *child*.GetNextMatch();

Make *child* an assigned node;

said ShiftNextMatch() procedure comprises the steps:

Child --> GetNextUnassigned();

child.match_ptr++;

15 if (*child.match_ptr* == NULL), then;

child.match_ptr --> *child*.GetNextMatch();

wherein variables *head*, *head2*, and *child*, all correspond to concept nodes;

variable *queue* denotes a priority queue of the corresponding concept node; and

match_ptr is a pointer to the next possible match for a given concept node; Pop() is a

20 method to get the next node off the priority queue; Push() is a method to put a node on

the priority queue; Empty() is a method to check if the priority queue is empty; Copy() is

a method to copy a node; Complete() is a method to check if the children assignment is

complete; Valid() is a method to check if the children assignment meets the constraints;
and GetNextUnassigned() is a method to select a variable that is unassigned.

10. A matching algorithm comprising GetNextMatch(), AssignNextMatch(),
5 and ShiftNextMatch() procedures, wherein:

said GetNextMatch() procedure comprises the steps:

```
testqueue:    if queue.Empty();  
               return NULL;  
               head --> queue.Pop();  
               if head.Complete();  
               return head;  
               head2 --> head.Copy();  
               head2.AssignNextMatch();  
               if head2.Valid();
```

```
15         queue.Push(head2);  
         head.ShiftNextMatch();  
         queue.Push(head);
```

```
Goto testqueue;
```

said AssignNextMatch() procedure comprises the steps:

```
20         child --> GetNextUnassigned();  
         child.match_ptr++;  
         if (child.match_ptr == NULL), then;  
         child.match_ptr --> child.GetNextMatch();
```

Make *child* an assigned node;

said ShiftNextMatch() procedure comprises the steps:

Child --> GetNextUnassigned();

child.match_ptr++;

5 if (*child.match_ptr* == *NULL*), then;

child.match_ptr --> *child*.GetNextMatch();

wherein variables *head*, *head2*, and *child*, all correspond to concept nodes;

variable *queue* denotes a priority queue of the corresponding concept node; and

match_ptr is a pointer to the next possible match for a given concept node; Pop() is a

10 method to get the next node off the priority queue; Push() is a method to put a node on

the priority queue; Empty() is a method to check if the priority queue is empty; Copy() is

a method to copy a node; Complete() is a method to check if the children assignment is

complete; Valid() is a method to check if the children assignment meets the constraints;

and GetNextUnassigned() is a method to select a variable that is unassigned.

15 11. A program storage device readable by machine, tangibly embodying a
program of instructions executable by the machine to perform method steps for querying
stored multimedia data, said method steps comprising:

receiving into an intermediate level a high-level concept from a user

20 describing data to be retrieved;

translating, in said intermediate level, said high-level concept into
low-level queries by using system pre-defined high-level concepts;

transferring said low-level queries to a low level comprising one or more search engines; said one or more search engines performing a query of the stored multimedia information using said low-level queries.

- 5 12. The apparatus of claim 11 wherein said intermediate level comprises:
a set of library modules, said set of library modules comprising:
a concept library module for storing concepts; and
one or more library modules adapted to store said data from said one or more data sources;

10 a cataloger module adapted to construct a new concept from said high-level concept using data from said concept library and library modules, thereby creating a concept construct, and to pass said concept construct to said concept library module for storage as a concept; and

15 an interpreter module adapted to translate said high-level concept into low-level queries using said concepts stored in said construct library and to pass said low-level queries to said one or more search engines.

- 20 13. The apparatus of claim 12 wherein said set of library modules further comprises at least one library module selected from the group comprising:

a feature library module adapted to store multimedia features;
a matching algorithm library module adapted to store matching algorithms; and
a constraint library module adapted to store feature constraints.

14. The apparatus of claim 13 wherein each said library module further comprises an application program interface to receive said data from a said data source.

15. The apparatus of claim 13 wherein said cataloger module further performs the steps of:

selecting a set of concept features from said feature library module;
selecting a set of concepts from said concept library module for use as child-concepts; and
selecting a set of constraints on said child concepts from said constraint library module.

16. The apparatus of claim 11 wherein said each said concept comprises a triplet of a set of child-concepts, a set of features, and a set of relationships.

17. The apparatus of claim 16 wherein said concepts comprise a hierarchical fuzzy graph data tree-structure comprising nodes, aggregation edges, and association edges and wherein:

said nodes correspond to said concepts and said features;
said aggregation edges correspond to parent-child relationships; and
said association edges correspond to said constraints.

18. The apparatus of claim 17 wherein said edges are weighted.

19. The apparatus of claim 13 further comprising a matching algorithm comprising GetNextMatch(), AssignNextMatch(), and ShiftNextMatch() procedures, wherein:

said GetNextMatch() procedure comprises the steps:

5 testqueue: if *queue*.Empty();
 return *NULL*;
 head --> *queue*.Pop();
 if *head*.Complete();
 return *head*;
 head2 --> *head*.Copy();
 head2.AssignNextMatch();
 if *head2*.Valid();
 queue.Push(*head2*);
 head.ShiftNextMatch();
10 *queue*.Push(*head*);
15 Goto testqueue;

said AssignNextMatch() procedure comprises the steps:

child --> GetNextUnassigned();
 child.*match_ptr*++;
20 if (*child*.*match_ptr* == *NULL*), then;
 child.*match_ptr* --> *child*.GetNextMatch();

Make *child* an assigned node;

said ShiftNextMatch() procedure comprises the steps:

Child --> GetNextUnassigned();

child.match_ptr++;

if (*child.match_ptr* == *NULL*), then;

child.match_ptr --> *child*.GetNextMatch();

5 wherein variables *head*, *head2*, and *child*, all correspond to concept nodes;

variable *queue* denotes a priority queue of the corresponding concept node; and

match_ptr is a pointer to the next possible match for a given concept node; Pop() is a

method to get the next node off the priority queue; Push() is a method to put a node on

the priority queue; Empty() is a method to check if the priority queue is empty; Copy() is

10 a method to copy a node; Complete() is a method to check if the children assignment is

complete; Valid() is a method to check if the children assignment meets the constraints;

and GetNextUnassigned() is a method to select a variable that is unassigned.

106080 265266
0995397 080901